VŠB – TECHNICKÁ UNIVERZITA OSTRAVA
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zpracování zvukového signálu na počítači

Processing Sound Signal on Computer

2010 Tereza Moudrá

„Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně.
Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.“

V Ostravě ……. Podpis

Poděkování

Na tomto místě bych chtěla poděkovat vedoucímu své bakalářské práce panu doc. Ing. Ličevovi, CSc. za rady a připomínky k obsahu a formě zpracování.

Abstrakt

Ve své práci se zabývám problematikou optického rozpoznávání znaků. Výsledná aplikace se skládá ze dvou částí. V první části je vyřešen převod textu z obrázku na normální text a ve druhé části je tento text převeden do audio formy. Aplikace využívá pro převod na text freeware program Asprise OCR a pro zvukovou část používá aplikaci z bakalářské práce p. Brettšnajdra. Řešení předkládané bakalářské práce je rozděleno do několika úkonů v rámci jednoho celku tak, aby tato aplikace mohla sloužit zejména slabozrakým lidem, kteří kvůli svému postižení nemohou plnohodnotně získávat textové informace. Dále se tato bakalářská práce zabývá formátem .jpeg a okrajově seznamuje čtenáře s prací s grafickým uživatelským rozhraním v Javě.

Klíčová slova

Optické rozpoznání znaků, OCR, JPEG, JPG, Java, Asprise OCR

Abstract

In my thesis I deal with problems of optical character recognition. The final application consists of two parts. The first part is solved by the transfer of text from an image file to a normal text and in the second part the text is converted into audio form. The application uses freeware Asprise OCR for conversion to a text and for audio section it uses an application from the Bachelor thesis of Mr. Brettšnajdr. Solution of this thesis is divided into several tasks within a single unit so that this particular application could serve especially visually impaired people who cannot efficiently obtain text information due to their disability. This thesis also deals with jpeg format and marginally introduces the work with the graphical user interface in Java.

Keywords

Optical character recognition, OCR, JPEG, JPG, Java, Asprise OCR

Seznam použitých symbolů a zkratek

OCR	- Optical Character Recognition
JPEG	- Joint Photographics Experts Group
GUI	- Graphical User Interface
HTML	- HyperText Markup Language
PDF	- Portable Document Format

Obsah

Seznam obrázků

1. Úvod

Dnešní doba klade stále větší důraz na automatizaci ve všech různých odvětvích. Automatizace zrychluje a zjednodušuje celý proces, což je v moderní době velmi žádoucí. Tuto automatizaci obstarává často informační a výpočetní technika. Jedním z mnoha podoborů, snažících se o automatizaci je i tzv. optické rozpoznávání znaků neboli OCR (Optical Character Recognition). Tento podobor se zabývá problematikou rozpoznávání textu z obrazu a jeho různými metodami.

OCR můžeme využít i v běžném životě. Lidé, kteří trpí lehkou zrakovou vadou, toto optické rozpoznávání jistě plně využijí. Nabízí jim totiž komfort v podobě poskytnutí hodnotných informací, kterých nejsou schopni jiným způsobem dosáhnout. Jedná se o textové informace, které jsou špatně čitelné. Tito lidé mají poškozený zrak do té míry, že jim čtení dělá obtížné problémy a luštěním textu si ještě více namáhají už tak dost namáhané oči.

Cílem této bakalářské práce bylo vyvinout aplikaci, která by pomohla právě těmto lidem trpícím špatným zrakem a umožnit jim kvalitnější přísun textových informací. Umožnit jim tak, aby si mohli přečíst jakoukoli knihu, či jízdní řád. Aplikace je implementována v jazyce Java, neboť umožňuje nezávislost platformy a dovoluje tvorbu desktopové i mobilní aplikace.

Na základě zadání své bakalářské práce se nejprve seznamuji s technologií optického rozpoznávání, a to v kapitole Optické rozpoznávání znaků, kde rozebírám jednotlivé pojmy spjaté s tímto tématem. Dále také popisuji metody a principy rozpoznávání a také seznamuji s fonty písma, které jsou vhodné pro dokonalejší rozpoznávání. Dalším bodem v zadání bylo nastudování grafického formátu JPEG. Toto studium popisuji v kapitole Grafické formáty, kde je konkrétní formát popsán společně i s technikami jeho komprimace. V kapitole Volba softwarových prostředků je popsán programovací jazyk, který jsem použila pro svou aplikaci a dále také základy práce s grafickým uživatelským prostředím. Následující kapitola podrobněji popisuje funkce softwaru, který jsem pro svou práci využila. Poslední kapitolou je Návrh a realizace programu, kde detailněji vysvětluji svou volbu jednotlivých softwarů, programovacího jazyka a také dále popisuji více do hloubky jednotlivé funkce své aplikace.

2. Optické rozpoznávání znaků

2.1 Vysvětlení pojmu optického rozpoznávání znaků

Optické rozpoznávání znaků je často označováno jako OCR, což je zkratka anglického pojmenování Optical Character Recognition. Jedná se o proces, kdy se text nejčastěji v tištěné podobě převádí do počítačově zpracovatelné podoby. Předlohou může být strojem ale i rukou psaný text. OCR je obvykle součástí složitějšího procesu, jehož úkolem je převod dokumentu z papírové podoby do digitálního formátu a následné rozpoznání znaků. Digitalizaci lze provést prostřednictvím skeneru nebo digitálních fotoaparátů. Mnohdy bývá OCR program přímo součástí skenerů. Celý postup procesu je popsán níže.

2.2 Historie a využití OCR

Historii OCR můžeme datovat do doby kolem roku 1950. V polovině 50. let 20. století se OCR systémy staly komerčně dostupné. První opravdové OCR programy byly instalovány ve firmě Reader's Digest roku 1954. Tyto první systémy se užívaly pro převod ručně psaných záznamů. První verze OCR programu byly charakterizovány jednoduchým rozpoznáváním znaků. Časem se začaly rozvíjet a na trh se dostávaly programy s větší fontovou zásobou. V polovině 60. let a počátku 70. se začaly objevovat systémy, které dokázaly rozpoznat strojově tištěné i ručně psané texty. V této době vznikal známý systém IMB 1287, firma Toshiba vyvinula první automatický třídič dopisů podle poštovních čísel a v roce 1966 byl vytvořen Americký standart OCR znakové sady (OCR-A), který byl navržen tak, aby se dosáhlo co nejvyšší kvality optického rozpoznávání a zároveň byl dobře čitelný. Časem se vyvinul i Evropský font (OCR-B), který byl mnohem lépe čitelnější. V roce 1970 se začínaly vyvíjet programy, které dovedli rozpoznat i méně kvalitní dokumenty.

Hlavním využitím OCR systémů je rozhodně digitalizace dat a tedy i usnadnění práce. S OCR systémy se ale často setkáváme i v běžném životě a ani o tom nemusíme vědět. Například čtení kreditních karet se provádí na základě rozpoznání znaků napsaných na kartě nebo třeba čtením čárových kódů na výrobcích také využíváme OCR metody.

2.3 Princip OCR

Převod dokumentu do digitální podoby se skládá z několika částí. Postupně si všechny popíšeme.

2.3.1 Digitalizace dokumentu

Jestliže máme list papíru, ze kterého chceme text převést, nejprve musíme dokument digitalizovat. Prostředky, které k tomu používáme, jsou scannery nebo digitální fotoaparáty. Důležitými parametry pro výslednou kvalitu, a tudíž i pro úspěšnost správného převodu, je rozlišení a barevná hloubka. Minimum pro rozlišení je 300 DPI a u barevné hloubky postačí odstíny šedé. Jestliže bude v dokumentu i barevný obrázek, musí být nastavena větší barevná hloubka. Splněním těchto parametrů zaručíme vyšší pravděpodobnost, že převod znaků proběhne v pořádku. Jestliže bychom měli ale rozlišení příliš malé, obrysy písmen by se mohly zkreslovat nebo také splývat, lépe řečeno tedy dotýkat, a OCR program by jej poté nebyl schopný přečíst.

2.3.2 Obrazové úpravy

Dalším krokem je úprava obrazu. Může se stát, že se naskenuje stránka trochu na křivo. Toto se stává většinou, když se skenuje kniha po dvoustranách. Program by poté nebyl schopný text rozlišit, a proto se musí obraz narovnat pomocí algoritmů geometrické korekce. Některé OCR programy toto řeší automaticky. Jestliže se skenuje celá kniha po dvoustránkách, je lepší je rozřezat na jednotlivé.

Další úpravou může být ořez stránky, kdy se ořežou okraje, neboť nejsou potřeba a zbytečně by zpomalovaly proces a zabíraly místo na disku. Pokud je však podmínkou nechat výsledný text ve stejném formátu jako originál, není vhodné ořez aplikovat, protože by výstupem byl jen holý text. Ořez stránky je výhodný zejména, jestliže se zpracovává větší množství textu, neboť jsou vyšší nároky na diskovou kapacitu a také v dalších fázích procesu se nám práce výrazně urychlí.

Další úpravou je binarizace, což je převod obrazu na černobílý, čímž se většina šumu ztratí. Zároveň ale dochází ke ztrátě informací a obrazy znaků se stávají ostré a zubaté.

Dalšími úpravami je například změna jasu, kontrastu, sytosti, světlosti barev a tak dále.

2.3.3 Rozdělení dokumentu na bloky

Blok představuje odstavec, tabulku nebo obrázek. Má tvar obdélníku nebo útvaru, složeného z více obdélníků. Program pracuje s myšlenkou, že blok je oddělen od svého okolí prázdným místem, a proto dokáže snadno rozdělit celý dokument na jednotlivé bloky. Jestliže se v textu objeví i obrázek, stane se z něj samostatný blok. Na rozdíl však od textových bloků se již dále nezpracovává, popřípadě jen změní barvu (z barevné na černobílou, pokud jsme zadávali

barevnou hloubku jako odstíny šedé). Při hledání bloků by však mohlo dojít k problému, že každý řádek nakonec skončí jako samostatný blok. Tento problém však řeší skutečnost, že řádky jsou od sebe vzdáleny menší mezerou než jednotlivé bloky.

2.3.4 Rozpoznání bloků

V tomto kroku následuje rozpoznávání v jednotlivých blocích. Každý blok je rozdělen na horizontální pásy. Tyto pásy se mohou částečně překrývat. Následně se rozpoznávají jednotlivé znaky. Tyto znaky se poté porovnávají se vzory znaků, a jestliže znak se vzorem souhlasí, zapíše se hodnota přiřazená ke konkrétnímu vzoru. Je důležité zadat informaci o jazyku dokumentu, aby OCR program věděl, který jazyk pro převod použít.

2.3.5 Lexikální postprocessing

Jedná se o proces, kdy probíhá korekce převedeného textu. Proces probíhá automaticky. Když však program narazí na nějaké špatně rozpoznatelné slovo nebo na slovo, které vůbec nezná, nechá rozhodnout uživatele. Jemu se zobrazí pokus o převod textu i výřez originálního dokumentu, kde se příslušné slovo nachází. Jestliže program nezná uživatelem zadané slovo, uloží si jej do paměti. Program se může tedy i přiučit novým slovům. Program se snaží zkontrolovat i souslednost slov a je obdobou korekce pravopisu například v Microsoft Word. Vždy je ale lepší konečný text zkontrolovat manuálně, neboť se vždy může stát, že program nepřeloží nějaké znaky správně.

2.3.6 Uložení ve zvoleném formátu

Existuje mnoho formátů, které můžeme použít jako výstupní formát. Nejčastějšími formáty jsou: HTML, PDF, Microsoft Word, Open Document a jiné.

2.4 Metody OCR

Existují dva základní druhy rozpoznávání. Těmi jsou porovnávání matic a extrakce příznaků. Z těchto dvou možností, jak rozeznat znaky, je porovnávání matic jednodušší a běžnější. Tyto metody se provádí ve fázi, kdy je již celý dokument obrazově upravený a rozčleněný na jednotlivé bloky i řádky. Informace o metodách OCR jsem čerpala z internetových článků viz. [1] a [2].

2.4.1 Porovnání matic

Jejím úkolem je rozložení každého znaku do rastrové matice. Tato matice se poté bod po bodu porovnává se vzory. Vybrán je znak, který nejvíce odpovídá porovnání matice se vzorem.

Tato metoda je snadno implementovatelná, avšak je velmi citlivá na jakýkoli šum v předloze, neboť sebemenší tečku považuje za součást znaku.

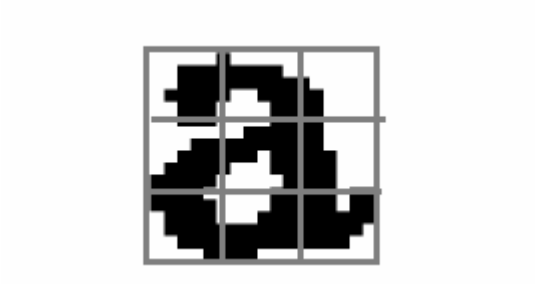
2.4.2 Extrakce příznaků

Jedná se o metodu, kde neplatí striktní porovnávání s předepsaným vzorem. Také se může nazývat inteligentní rozpoznávání znaků (Intelligent Character Recognition - ICR) nebo topologická funkce analýzy (Topological Feature Analysis). Tato metoda má mnoho verzí, které se od sebe odlišují mírou použité „inteligence“, což má za následek vyšší či nižší výkonnost. Na rozdíl od předchozí metody, je tato všestrannější a využívá se tehdy, jestliže jsou znaky méně předvídatelné. Techniku extrakce symbolu lze rozdělit do tří hlavních skupin:

- rozčlenění bodů
- transformace a sériová expanze
- strukturální analýza

Rozčlenění bodů

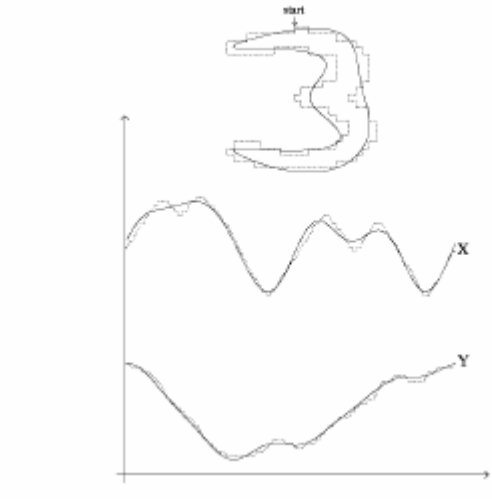
Tento typ techniky je založen na statickém rozložení bodů. Tyto charakteristické rysy jsou tolerantní k pokřivení znaku nebo variabilitě stylu.



Obrázek 1 Rozčlenění bodů

Transformace a sériová expanze

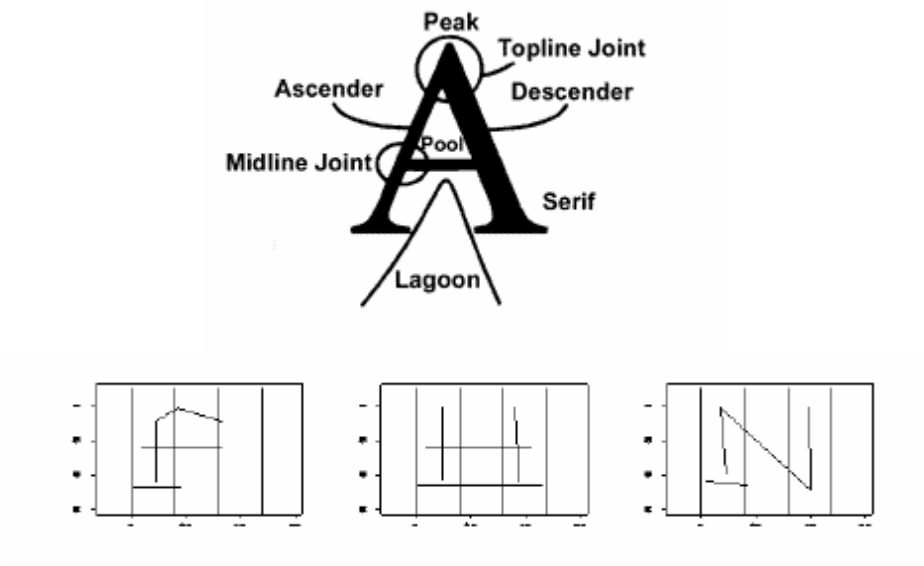
Tato technika využívá Fourierův, Haarův či Walshův transformační postup. Znak je transformován do křivky charakterizující nejdůležitější rysy daného znaku. Má vysokou citlivost na šum postihující okraje znaku. Příkladem jsou třeba chybné mezery v symbolu.



Obrázek 2 Transformace znaku

Strukturální analýza

Tato technika rozebírá a popisuje každý znak geometricky a topologicky. Každý symbol je rozebrán na určující části a ty jsou topologicky porovnávány s originálem. Tato technika velice toleruje šum i variabilitu stylů, avšak není jednoduše implementovatelná.



Obrázek 3 Geometrický způsob rozložení a popsání znaku

2.5 OCR fonty

Na začátku rozvoje optického rozpoznávání znaků bylo zapotřebí vytvoření jednotného typu písma, které by každý počítač snadno rozpoznal. Počítačová editace textu totiž nabízela a dosud nabízí velké množství různých typů písma, takzvaných fontů. OCR program by tedy musel mít kompletní databázi všech fontů, což by bylo náročné na paměť a téměř i nemožné, neboť se

stále vyvíjely další a další fonty. Vzniklo tedy několik fontů, určených přímo pro optické rozpoznávání znaků. Nejznámějšími fonty jsou OCR-A a OCR-B.

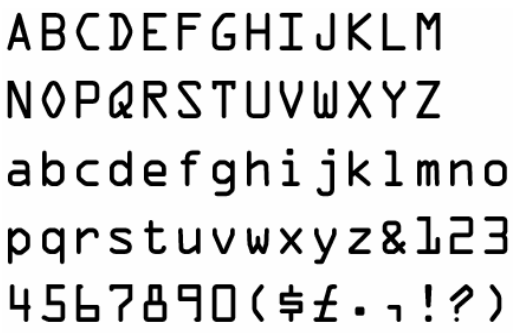
2.5.1 OCR-A

Byl standardizován institucí American National Standards Institute (ANSI) jako X3.4-1977. Později byl přejmenován na ISO 1073-1:1976. K dispozici je i německý standard pro OCR-A, nazýván DIN 66008. Všechny tyto normy jsou chráněny autorským právem.

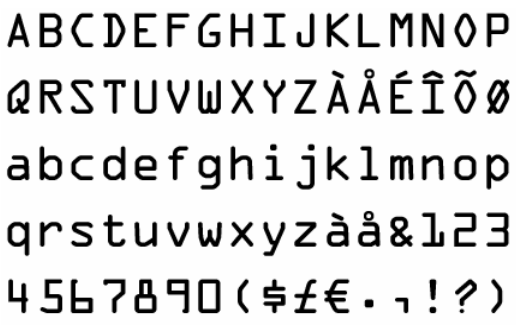
O vývoj OCR-A se zasloužila organizace American Type Founders a OCR-A se stal jedním z prvních typů písma pro optické rozpoznávání, které splňovalo kritéria stanovená americkými úřady pro normalizaci. Vzhled písma je jednoduchý, takže jej lze snadno přečíst strojově, ale je mnohem obtížnější pro lidské oko. Má silné tahy, je neproporcionální a mezi jednotlivými znaky jsou mezery v rozmezí 0,23 cm – 0,46 cm.

Přestože technologie optického rozpoznávání znaků dospěla do bodu, kdy tyto jednoduché fonty již nejsou nutné, OCR-A zůstává stále v provozu. Některé společnosti dosud využívají tyto fonty například pro číslo účtu. Navíc, někteří lidé stále dávají přednost tomuto fontu pro svůj jedinečný styl.

Dalším typem je OCR-A Extended. Jak už z názvu vyplývá, jde o rozšířenou verzi OCR-A. Tento font je součástí základních fontů pro Microsoft Word 2007.



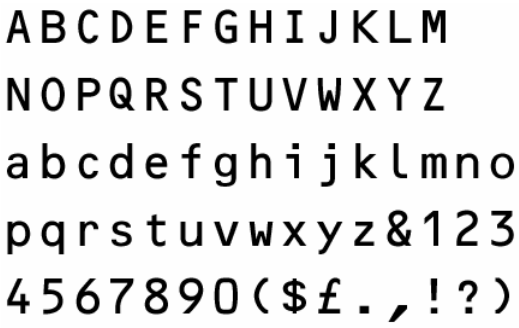
Obrázek 4 ukázka fontu OCR-A



Obrázek 5 ukázka fontu OCR-A Extended

2.5.2 OCR-B

Je to sada neproporciálních písmen, vyvinutých v roce 1968 Adrianem Frutigerem pro Monotype podle pokynů výrobce European Computer Manufacture's Association standard. Jeho hlavní funkcí bylo usnadnění operací optického rozpoznávání znaků. Font OCR-B byl standardizován v roce 1973. Zahrnuje všechny ASCII znaky a jiné symboly, které se mohou vyskytovat v bankovnictví. Od OCR-A se liší zejména ve vzhledu. Čtení OCR-B je pro lidské oko snadnější a má méně „technický" vzhled.



Obrázek 6 ukázka fontu OCR-B

3. Grafické formáty

Výsledná aplikace pracuje s grafickými soubory. Tato kapitola detailněji popisuje jeden z grafických formátů.

3.1 JPEG

Tato zkratka znamená Joint Photographics Experts Group a jedná se o standardní metodu pro komprimaci počítačových obrázků s kompresí ve fotorealistické podobě. Stejným názvem je ale označován i přímo formát souboru, který tuto metodu používá. Nejrozšířenějšími příponami jsou .jpg, .jpeg, .jpe. Avšak skutečný název typu souboru je JFIF (JPEG File Interchange Format), neboť JPEG je název metody, která pouze předepisuje způsob ztrátové i bezeztrátové komprese rastrových obrázků. Stupeň kvality komprimace se definuje pomocí Q-faktoru (Quality factor), u kterého platí, že čím nižší Q-faktor je možné na obrázek aplikovat, tím vyšší bude komprimační poměr, avšak cenou větších ztrát. Postupem času však název JPEG zlidověl a nyní se používá jak pro označení komprimační metody, tak i jako formát, který tuto metodu využívá. Jako formát se používá nejčastěji pro přenášení a ukládání fotografií na internetu, neboť má relativně malou velikost. U JPEGu je možné dosáhnout komprimačního poměru 1:50 až 1:100 při zanedbatelné ztrátě informace. JPEG není a ani nikdy nebyl určen pro ukládání obrázků obsahujících malé množství barev, kontrastní barevné přechody, ostré hrany nebo písmo. JPEG byl vyvinut zejména pro ukládání fotografií, naskenovaných dokumentů či rentgenových a ultrazvukových snímků. Podrobnější informace lze najít v literatuře [3].

3.2 Bezeztrátové komprimační metody

Jsou to metody, kdy ukládaný rastrový obraz není v žádném případě změněn natolik, aby došlo ke ztrátě informace. Mění se pouze celkový počet barev u formátů nepodporující plnobarevné režimy. Některé grafické formáty nenabízí žádnou komprimační metodu. Příkladem může být například PBM, PGM, PPM, PAM. Jiné využívají pouze jednoduché kódování RLE nebo metody LZ77 a LZW (GIF a PNG).

S rozvojem počítačové grafiky začaly počítače více využívat zpracování fotografií, jejich archivaci nebo prezentaci. Velkého významu nabýval i Internet, kde se postupem doby stále více využívaly rastrové grafické soubory, které měly mnohdy před svou komprimací obrovskou velikost. Začaly se tedy hledat způsoby, jak tyto soubory zmenšit. Již počátkem osmdesátých let

minulého století se proto začaly hledat vhodné algoritmy určené speciálně pro ukládání plnobarevných rastrových obrázků.

3.3 Ztrátové komprimační metody

Tyto metody vycházejí z jednoduchého principu. Informaci, která je sice v obrázku prokazatelně přítomna, ale člověk ji může postrádat, je možné selektivně odstranit a dosáhnout tak výrazného komprimačního poměru. S JPEGem můžeme dosáhnout stavu, kdy je jeden pixel v komprimovaném obrázku popsán pouze jedním bitem, zatímco původní obrázek má 24 bitů na pixel.

Algoritmus ztrátové komprese se rozděluje do dvou důležitých částí: transformace původních dat a potlačení různě důležitých dat. Transformace využívá některé z transformačních metod (DCT, FFT, DWT). Tyto transformace převedou původní data do jiných domén. Většina informací je pak uchována v mnohem menším objemu než původně. Potlačení některých dat poté rozhoduje o tom, jaká data mohou být potlačena nebo dokonce úplně odstraněna. Při kompresi se tedy posuzuje, které frekvence v obrazu jsou důležité, aby člověk na obrázku viděl to, co na něm vidět má.

3.4 Metody komprese

Hlavním důvodem, proč komprimovat soubory, je zmenšení jejich objemu, ale přitom musí být zachovány veškeré informace, které se nacházejí u původních údajů. Musí však také ale platit zákon zachování informace. Základní myšlenkou komprese je odstranění redundance ze souboru. Existují dva typy komprimace - bezeztrátová a ztrátová komprese.

Do kategorie bezeztrátové komprese patří například metoda RLE - Run Length Encoding. Jedná se o nejjednodušší metodu, která je založena na předpokladu, že obrázek se skládá z oblastí, které jsou vyplněny určitou charakteristickou barvou. Pixely, následující po sobě a mající stejnou barvu, jsou tedy zakódovány kombinací početnosti znaku a samotného znaku. Další metodou bezeztrátové komprese je LZW - Lempel-Ziv a Welch či Huffmanovo kódování, jehož hlavní myšlenkou je vytvoření prefixového kódu minimální délky ze statistiky komprimovaného textu.

Do kategorie ztrátové komprese se řadí algoritmus DCT. Tento algoritmus funguje na principu kosinusové transformace a kvantizaci kmitočtových koeficientů. Podrobnější informace o metodách komprese lze získat v knize uvedené v použité literatuře viz. [4].

4. Volba softwarových prostředků

Pro tuto aplikaci jsem si vybrala programovací jazyk Java. Je to programovací jazyk vyvinutý firmou Sun Microsystems. Jde o objektově orientovaný jazyk vycházející z programovacího jazyka C++. Java odstraňuje nedostatky tohoto jazyka a navíc přidává plno dalších užitečných vlastností. Například již nepracuje s ukazateli, ale namísto toho s referencemi. Dále je implementován mechanismus vláken, lze tedy spustit více úloh v rámci jednoho programu současně. Java také pracuje s výjimkami, což zaručuje odchycení i zpracování runtime chyb. K dispozici jsou obsáhlé knihovny, pomocí nichž lze pracovat například s textem, s komprimovanými soubory, lze komunikovat s SQL databázemi či vytvářet grafické uživatelské rozhraní. Tento jazyk lze také samozřejmě dále rozšiřovat pomocí pluginů. Výhodou Javy je také její hardwarová nezávislost, jelikož je překládána do speciálního mezikódu – tzv. bytecode, který je na konkrétním počítači nebo zařízení překládán do nativního kódu. Javovský program, napsaný na PC pod operačním systémem Windows, lze tedy spustit i na PC s Linuxem.

JRE je zkratka z anglického názvu Java Runtime Environment, která označuje prostředí pro běh programů v Javě. Toto prostředí zahrnuje interpret Javy a standardní knihovny. JDK je Java Development Kit a obsahuje JRE, překladač a další vývojové nástroje. Java má několik verzí a jsou jimi:

- Java SE – Standard Edition
- Java EE – Enterprise Edition
- Java ME – Micro Edition

Java EE rozšiřuje Java SE a je určena pro psaní webových aplikací. Oproti tomu Java ME je určena speciálně pro psaní aplikací do mobilních zařízení, jako jsou telefony či PDA. Více informací o tomto programovacím jazyce lze nalézt v literatuře [5].

4.1 Grafické uživatelské rozhraní - GUI

První verzí grafiky v Javě byla knihovna nazvaná Abstract Windowing Toolkit (AWT). Vyskytovala se již v JDK 1.0. Hlavní myšlenkou AWT knihovny bylo, že každá komponenta měla svůj nativní protějšek v systému. I když se jednalo o platformově nezávislé rozhraní, přenositelnost byla problematická, jelikož existovaly rozdílné vlastnosti nativní úrovně GUI.

AWT knihovny tvoří následující balíky java.awt, java.awt.event, java.awt.image, java.awt.datatransfer.

Později, a to ve verzi 1.2, vývojáři Javy vyvinuli novou grafickou knihovnu nazvanou Java Foundation Classes (JFC), známější však pod názvem SWING. Tato knihovna jen nepatrně vycházela z původní koncepce knihovny AWT, ale její základní vlastnosti byly vytvořeny zcela nově. Knihovna SWING je kompletně implementována v Javě, což znamená, že GUI není vytvořenou na nativních komponentech a je tedy zcela platformově nezávislá. SWING nabízí možnost využití dědičnosti a kompozice. Jednotlivé komponenty využívají objektové vlastnosti Javy. Také se velmi dbá na oddělení funkcionality od vzhledu, tzv. look & feel.

Další GUI knihovnou je Standard Widget Toolkit neboli SWT. Tato knihovna měla za úkol odstraňovat nedostatky SWING, a to zejména problémy s nároky na rychlosti procesoru a paměť. Avšak ve výsledku nedochází k téměř žádnému snížení a je zhruba stejně náročná jako SWING.

Základním prvkem grafického uživatelského rozhraní jsou komponenty. Jsou to jednotlivé prvky, se kterými můžeme pracovat. Jde například o textové pole, label, tlačítko, tabulka, menu a mnoho dalšího. Komponenty lze spojovat pomocí Listu, což je soubor komponent, které nelze překrýt. Existuje i jiný soubor komponent, ten se nazývá kontejner. Příkladem je třeba Frame, Panel, či MenuBar.

Důležitým parametrem při tvorbě GUI je pozice jednotlivých prvků. Java poskytuje třídu, která umožňuje uspořádat komponenty obsažené v kontejneru do podoby, při které není nutné zadávat absolutní pozici komponent. K tomu složí schémata Layout Managers. Existuje několik základních typů:

- FlowLayout – rozmístění komponent do řady za sebou,
- GridLayout – umístění komponenty do mřížky,
- BorderLayout – rozmístění komponent na základě světových stran (north, south, west, east, center).

Jednotlivé typy Layout Managers lze samozřejmě libovolně kombinovat k dosažení požadovaného vzhledu.

Nejdůležitější částí tvorby grafického uživatelského rozhraní není ani tak vzhled aplikace, ale spíše reakce jednotlivých prvků na požadavky uživatele. Interakce mezi uživatelem a vizuálními prvky je vyřešena pomocí událostí (events). Události fungují na základě zaslání informace o změně stavu zdrojového objektu všem objektům, které o to požádají. Těmto

objektům se říká Listeners (posluchači). K tomu, aby se mohl objekt stát posluchačem pro příslušnou zprávu, stačí definovat metodu, která bude na zprávu o události reagovat. Tyto metody jsou poté sdruženy do rozhraní, které je potomkem rozhraní java.util.EventListener. GUI nabízí již existující třídy zpráv a jim odpovídající rozhraní pro posluchače:

- ActionEvent – nastane při změně stavu komponenty (např. stisk tlačítka),
- KeyEvent – při změně stisku klávesy,
- MouseEvent – při změně stavu myši,
- WindowsEvent – při práci s oknem.

Java obsahuje také třídu java.awt.Graphics, která umožňuje práci s kreslením základních grafických prvků, jako jsou úsečka, obdélník, kružnice, výpis textu v grafickém režimu a vykreslování rastrového obrazu. Pro použití této třídy je nutné vytvořit její instanci příkazem:

```
panel.getGraphics();
```

Grafický kontext lze přiřadit každé komponentě, avšak nejčastěji se pro grafický výstup využívá komponenta java.awt.Canvas.

Java podporuje práci s rastrovými obrázky formátu GIF, JPEG a PNG. K jejich zpracování lze využít speciální třídy java.awt.Image. Jde však pouze o abstraktní třídu, a proto musíme k načtení obrázku využít třídu public abstract class Toolkit, která vytvoří propojení mezi komponentami a jejich implementací.

```
Image myImage;
myImage = Toolkit.getDefaultToolkit().getImage("picture.jpeg");
```

5. Aplikace pro převod pomocí OCR

Výsledná aplikace využívá pro převod pomocí metody OCR volně stažitelný program, který se nazývá Asprise OCR.

5.1 Instalace OCR

Nutností, pro úspěšné rozběhnutí této aplikace, je mít nainstalovanou Javu verze 1.2 nebo vyšší. Abychom měli přístup k balíčku com.asprise.util.ocr a tedy i jeho metodám, musíme nastavit aspriseOCR.jar jako classpath. To můžeme učinit přímo ve vývojovém prostředí NetBeans přidáním do vytvářeného projektu pod položkou Properties -> Libraries -> Compile. Dále také potřebujeme nativní knihovnu AspriseOCR.dll. Tento soubor vložíme do systémového adresáře C:/Windows/System32.

5.2 Využití OCR

Před samotným použitím musíme nejprve importovat balíček aplikace Asprise OCR přímo do našeho projektu. To zařídíme pomocí kódu:

```
import com.asprise.util.ocr.OCR
```

Poté vytvoříme proměnnou typu BufferedImage a vložíme do ní obrázek na lokálním disku.

```
BufferedImage image = ImageIO.read(new File("ocr.jpeg"));
```

Když máme načtený obrázek, můžeme přistoupit k samotnému rozpoznávání. Vytvoříme si tedy proměnnou typu String, do které vložíme výsledek volané metody. Ke knihovně Asprise OCR a jejím metodám přistoupíme následovně:

```
String s = new OCR().recognizeAll(image);
```

Z aplikace Asprise OCR lze využít několik typů metod. Jednotlivé metody jsou popsány níže.

5.2.1 Rozpoznávání znaků z obrázku

Tato metoda rozpozná pouze písmena a číslice, čárové kódy jsou zde přehlíženy. Implementuje se následujícím způsobem:

```
String s = new OCR().recognizeCharacters(image);
```

5.2.2 Rozpoznávání znaků z části obrázku

Tato metoda rozpozná pouze znaky, které jsou obsaženy v části obrázku, kterou jsme definovali. Dá se říci, že při této metodě využíváme „virtuálně oříznutý“ obrázek. Této metody lze efektivně využít například tehdy, jestliže rozpoznáváme skenovanou knihu a všechny strany mají stejně velký okraj, který nemusí program procházet a zbytečně ztrácet čas. Implementace vypadá následovně:

```
BufferedImage image = ImageIO.read(file);
image = image.getSubimage(0, 0, 200, 100);
String s = new OCR().recognizeEverything(image);
```

5.2.3 Explicitní nastavení nativní knihovny

Pomocí této metody lze nastavit explicitně cestu k místu uložení nativní knihovny. Jestliže nastavíme knihovnu touto cestou, předchozí nastavení nebude bráno v potaz.

```
OCR.setLibraryPath("C:/tmp/AspriseOCR.dll");
```

Další metody se týkají pouze rozpoznávání čárových kódů, a proto zde již nebudou uváděny, neboť má práce se zabývá pouze rozpoznáváním znaků.

6. Návrh a realizace aplikace

Má bakalářská práce se zabývá OCR problematikou a výsledkem mé studie by měla vzejít funkční aplikace, která by umožňovala překlad obrázku na text a následné převedení do zvukové podoby. Převedení obrazu na text zprostředkovává aplikace AspriseOCR, lépe řečeno tedy pouze knihovna tohoto produktu. Tento software ale umí pracovat pouze s texty bez diakritiky mimo písmena s čárkou, a proto ke správnému překladu českého textu je zapotřebí ještě dodatečná korektura. Tuto kontrolu můžeme provést strojově nebo jen manuálně tak, že vlastnoručně přepíšeme špatně převedené znaky. Další částí mé aplikace je přečtení textu počítačem. Využila jsem bakalářské práce p. Brettšnajdra, zabývající se syntézou zvukového signálu a jeho zdrojový kód jsem zakomponovala do své aplikace. Díky tomu tedy lze převedený text převést do zvukové podoby.

6.1 Volba softwarových prostředků

Pro svou práci jsem zvolila již hotový software pro rozpoznávání znaků, neboť vývoj této techniky by bylo značně rozsáhlé. Existuje mnoho produktů, které se zabývají prací s OCR problematikou. Avšak mnohé z nich jsou pouze komerční a mnoho z těch, které nejsou placené, jsou místo toho velmi složité na implementaci do vlastní aplikace nebo jsou napsány v jiném programovacím jazyce, než je Java.

Proto jsem využila programu AspriseOCR. Tento program je volně k dispozici na internetu *http://asprise.com* jako freeware. Program nabízí knihovnu metod, které umožňují rozpoznávat znaky. Jeho největší předností a také i důvod, proč jsem si tento program vybrala, spočívá v tom, že je vytvořen pomocí programovacího jazyku Java a nabízí podrobnou dokumentaci ke své knihovně a tudíž i snadnou implementaci do vlastní Java aplikace. Samotná instalace spočívá jen v několik jednoduchých krocích. viz 5.1.

6.2 Volba typu aplikace

Mým zadáním bakalářské práce bylo vytvoření aplikace pro mobilní zařízení. Mobilní aplikace se vyvíjí pomocí jazyka Java Micro Edition (Java ME). Tato edice je velmi zjednodušenou verzí klasické Javy pro desktopové aplikace (Java Standard Edition - Java SE) a mnoho pomocných tříd a knihoven, které bychom ve standardní edici našli, je zde zcela vypuštěno. Důvodem absence mnohých tříd a knihoven je fakt, že výkon a kapacita mobilních

zařízení jsou značně omezeny. Tudíž nemůžeme očekávat, že aplikace, které fungují ve standardní edici, budou fungovat i ve verzi pro mobilní zařízení, jelikož mnohé z nich využívají doplňkové knihovny či třídy, které Java pro mobilní zařízení nepodporuje. Přestože zadáním bakalářské práce bylo vypracovat mobilní aplikaci, musela jsem se uchýlit k desktopové aplikaci. Důvodem této volby bylo zjištění, že se na trhu v této chvíli nenachází žádná vhodná OCR aplikace bez poplatků, kterou by bylo možno implementovat do mobilních aplikací. Podařilo se mi najít pouze OCR aplikace, které nebyly vyvinuty speciálně pro mobilní zařízení a tudíž ani nebyly tímto typem zařízení podporovány. Příkladem je využitá aplikace Asprise OCR.

Implementovat nelze ani aplikaci p. Brettšnajdra pro převod na zvuk, jelikož využívá knihovny, které v mobilních aplikacích nelze používat. Při důkladném prohledání nabídky dostupných aplikací zabývajících se převodem textu na zvuk, jsem shledala veškeré aplikace za nevhodné ze stejného důvodu jako OCR aplikace.

6.3 Korektura

Jak již bylo zmíněno výše, metoda OCR umí pracovat pouze s anglicky psaným textem. Tudíž špatně rozpoznává český text s diakritikou. Kvůli tomuto důvodu byla naimplementována metoda provádějící korekturu. Tato strojová korektura se provádí velice jednoduchým způsobem. Aplikace převádí nerozpoznatelné znaky, v našem případě tedy znaky s diakritikou do unikátního kódu a tedy je velice snadné poté tyto kódy nahradit patřičnými znaky. Nejprve jsem si tedy vypsala veškeré znaky české abecedy, uložila je ve formátu .jpg a poté převedla na text pomocí své aplikace. Tím vznikla překladová tabulka, díky které jsem dále byla schopná přiřadit patřičnému kódu jeho znak. Aplikace si umí poradit s písmeny s čárkou, ale znaky s háčky nerozezná. Místo nich do textu vloží speciální kódy viz.tabulka 1.

znak	znak rozpoznaný aplikací
ě	\code(011b)
š	\code(0161)
č	\code(010d)
ř	\code(01d0)
ů	\code(016f)
ř	\code(012d)
ě	\code(0115)

Tabulka 1 ukázka překladové tabulky

Jakmile je strojová korektura hotová, nejlepší volbou je ještě konečná manuální kontrola, kdy je možné v textu přepisovat špatné znaky a porovnávat nesrovnalosti s originální předlohou.

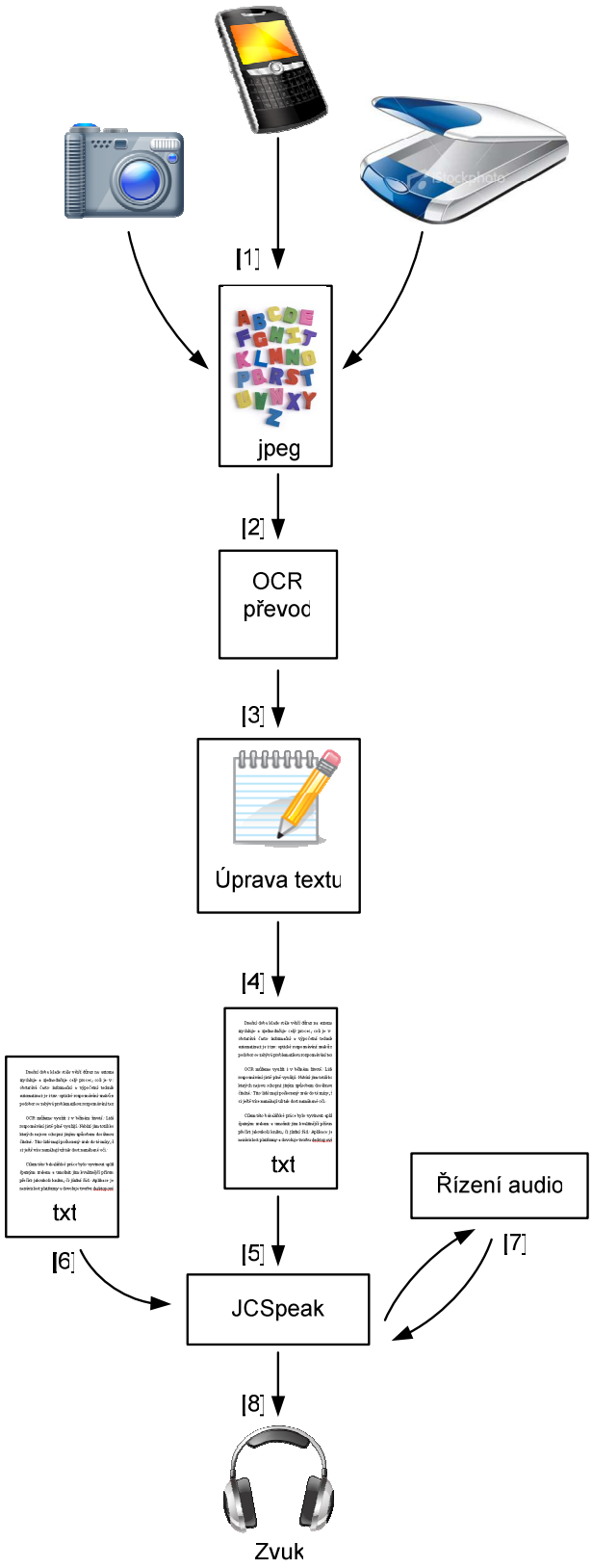
Jelikož je tato aplikace věnována zejména lidem postiženým zrakovou vadou, součástí aplikace je i seznam zkratek, které by se v textu eventuálně mohly objevit. Při čtení by dělalo značné problémy pochopení významu jednotlivých zkratek, jelikož by byly přečteny písmeno po písmenu. Příkladem je třeba zkratka PC. Každý by ji přečetl [pí sí], avšak program by přečetl jen [p c], což by bylo pro někoho, kdo špatně vidí, velmi nesrozumitelné. Jestliže by se jednalo ještě o mnohem složitější a delší slovo, bylo by to velmi matoucí. Proto aplikace využívá textového souboru, ve kterém jsou uloženy zkratky a jejich pravý význam. Při korektuře pak aplikace automaticky tyto zkratky nahradí a v textu se již objeví celá slova. V našem případě by tedy program převedl zkratku PC na slovo počítač. Seznam je ukládán do textového souboru, aby jej bylo možno doplňovat o nová slovíčka.

6.4 Zvuk

Aplikace má především sloužit lidem, kteří špatně vidí text, a tudíž se jim s obtížemi čtou jednotlivá písmena. Pomocí této aplikace načteme obraz, jímž může být například vyfocená stránka knihy či časopisu, poté jej převedeme na text a ten necháme těmto lidem přečíst. Zvukovou část zajišťuje kód p. Brettšnajdra, který se ve své bakalářské práci zajímal o syntézu řeči v českém jazyce. Jeho zdrojový kód obsahuje analyzátor i syntetizátor elementárních jednotek řeči. Tato bakalářská práce obsahuje mnohem více funkcí, avšak já využívám pouze ozvučení textu. Podrobnější informace ohledně principu této práce naleznete zde [7].

6.5 Datové toky

Zdrojem získání dat může být digitální fotoaparát, mobilní telefon vybavený digitálním fotoaparátem či skener. Výsledný soubor v tomto kroku je uložen ve formátu .jpeg nebo .jpg (popřípadě .bmp) viz obr. 6.1. první bod. Ve druhém bodě se získaný soubor převádí pomocí OCR metody na text, než však dojde k uložení ve formátu .txt, musí proběhnout dodatečná kontrola textu a oprava případných chyb. Korektura je znázorněna na schématu v bodě 3. Ve čtvrtém bodě vzniká již kompletně převedený text a může dojít k uložení na lokální disk. Bod 5 znázorňuje další cestu textového souboru, a to do aplikace JCSpeak, kterou vytvořil p. Brettšnajdr v rámci své bakalářské práce. Do aplikace JCSpeak lze vkládat i obyčejný textový soubor, který jsme si vytvořili jiným způsobem, než je převod pomocí OCR metody (bod 6). Program JCSpeak nejprve zpracuje text k převodu do audio formy a poté jej následně provede (bod 7). Výsledkem je přečtení převedeného či nahraného textu (bod 8).



Obrázek 7 Schéma datových toků

6.6 Architektura aplikace

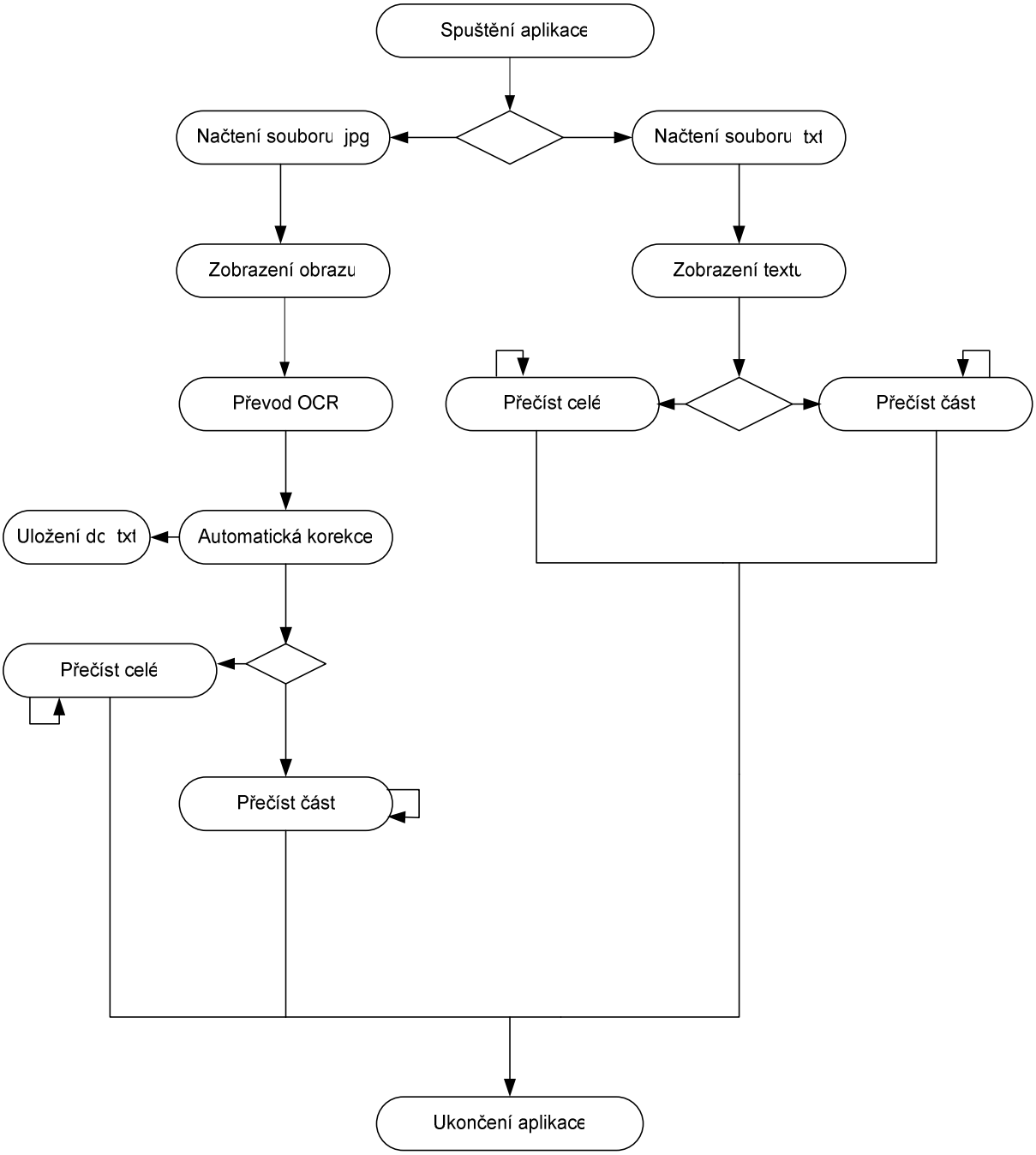
Celý proces probíhá v několika krocích. Spuštěním aplikace se nám otevře okno s nabídkou načtení obrazového či textového souboru.

Jestliže došlo k načtení obrázku, dalším krokem je vykreslení fotografie do okna. Obraz je automaticky zmenšen na velikost náhledu. Jestliže došlo ke zmenšení, zobrazí se nám nabídka pro dodatečné zvětšování a zmenšování obrazu.

Následným krokem je převod obrazu na text díky optického rozpoznávání za pomocí freeware Asprise OCR.

Aplikace Asprise OCR nezná český jazyk, tudíž znaky s čárkami a háčky nerozezná. Proto je dalším krokem automatická korekce, která převede špatně rozpoznané znaky na korektní. Je umožněna i dodatečná ruční korektura přepnutím textu do stavu editace. Výsledný text lze uložit do textového souboru formátu .txt.

Další možností je daný text nechat přečíst. Lze přečíst celý text, jen označený text nebo nechat aplikaci přečíst text větu po větě s možností vždy se vrátit k předchozí větě. Ke „čtecí“ fázi se lze dostat i tím způsobem, že na začátku načteme textový soubor. Ten se nám rovnou uloží do pole, vyhrazeného pro text a následně tedy můžeme text převést do audio formy se stejnými možnostmi jako při čtení textu převedeného z obrázku.



Obrázek 8 Architektura aplikace

6.7 Doplňující funkce

Dvěma základními funkcemi je jistě převod z obrazu na text a následné čtení. Součástí aplikace je ale i režim přehrávání věty po větě, což znamená, že můžeme přehrát libovolnou větu, kolikrát budeme chtít a navíc bude v naší plné režii, kdy se která věta spustí. Dalším způsobem čtení textu je přečtení aktuální věty. Aktuální větu označíme tak, že někde do věty umístíme kurzor. Číst můžeme také jen části vět či slov. Toho docílíme označením dané části textu a spuštěním čtení.

Dále je také pamatováno na uživatele, kterým je tato aplikace zejména věnována, a tudíž je umožněn výběr velikosti fontu. Také je možné celou aplikaci ovládat pomocí klávesových zkratek. Veškerý seznam těchto zkratek a jejich význam je uveden v příloze Uživatelský manuál.

Dodatečnou funkcí tohoto programu je elektronický jízdní řád. Tato funkce umožňuje zjišťovat dopravní spoje v rámci Dopravního podniku Ostrava. Aplikace je schopná zpracovávat html stránky nabízející právě zmiňovaný podnik, na kterých jsou jednotlivé dopravní linky. Pomocí regulárních výrazů se ze stránky získají jednotlivá data, která tato aplikace následně zpracovává. Aplikace umožňuje zobrazení seznamu stanic a jednotlivé časy odjezdů v závislosti na zvolené stanici. Dále také umožňuje jednoduché vyhledávání dopravních spojů. Vše lze virtuálně přečíst, popřípadě zopakovat. Ovládání je umožněno i pomocí klávesových zkratek. Html soubor lze načíst přímo z internetové stránky.

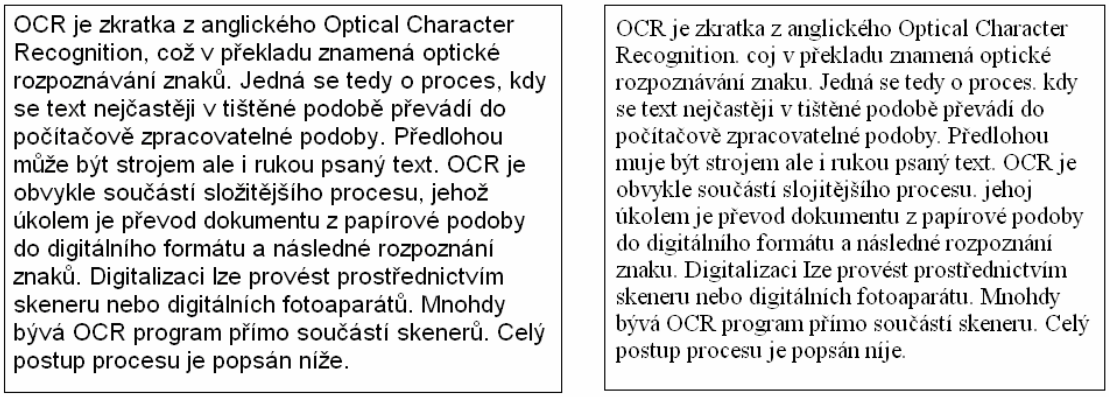
6.8 Testování

Vylepšenou aplikaci o finální korekci jsem podrobila testování úspěšnosti rozpoznávání. První testovaný dokument je tvořen přímo textem převedeným do formátu jpeg. Druhý dokument je již strana textu, vyfocená pomocí fotoaparátu. Třetí ukázka je zhotovena pomocí scanneru.

Prvním testovaným dokumentem je tedy soubor ve formátu .jpeg zhotovený pomocí grafického editoru. Vedle ukázky obrázku je text převeden pomocí mé aplikace viz obr. 9. Úspěšnost rozpoznání je 98,99%. Tato ukázka je na stejné bázi jako například screenshoot vytvořený z pdf souboru.

Počet znaků v textu: 594

Počet špatně převedených znaků: 6

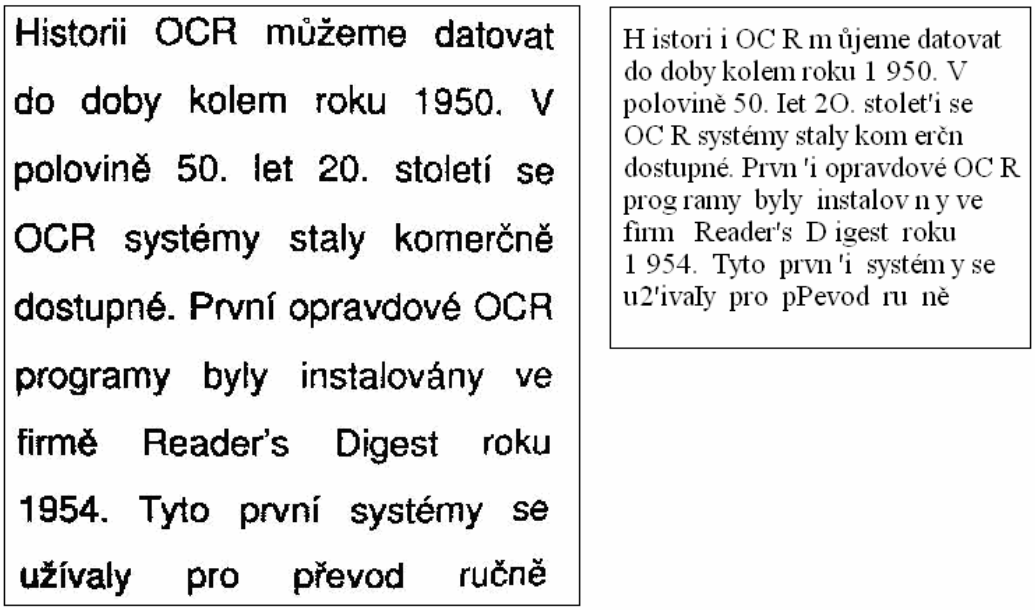


Obrázek 9 Test 1

Druhým testovaným dokumentem je soubor ve formátu .jpeg opatřený fotoaparátem s rozlišením 7,1 MPx viz obr. 10. Výsledný obraz je upraven v grafickém editoru a je na něj aplikováno barevné vyvážení a nastavení kontrastu a jasu. Úspěšnost rozpoznání je: 95,99%.

Počet znaků v textu: 299

Počet špatně převedených znaků: 12

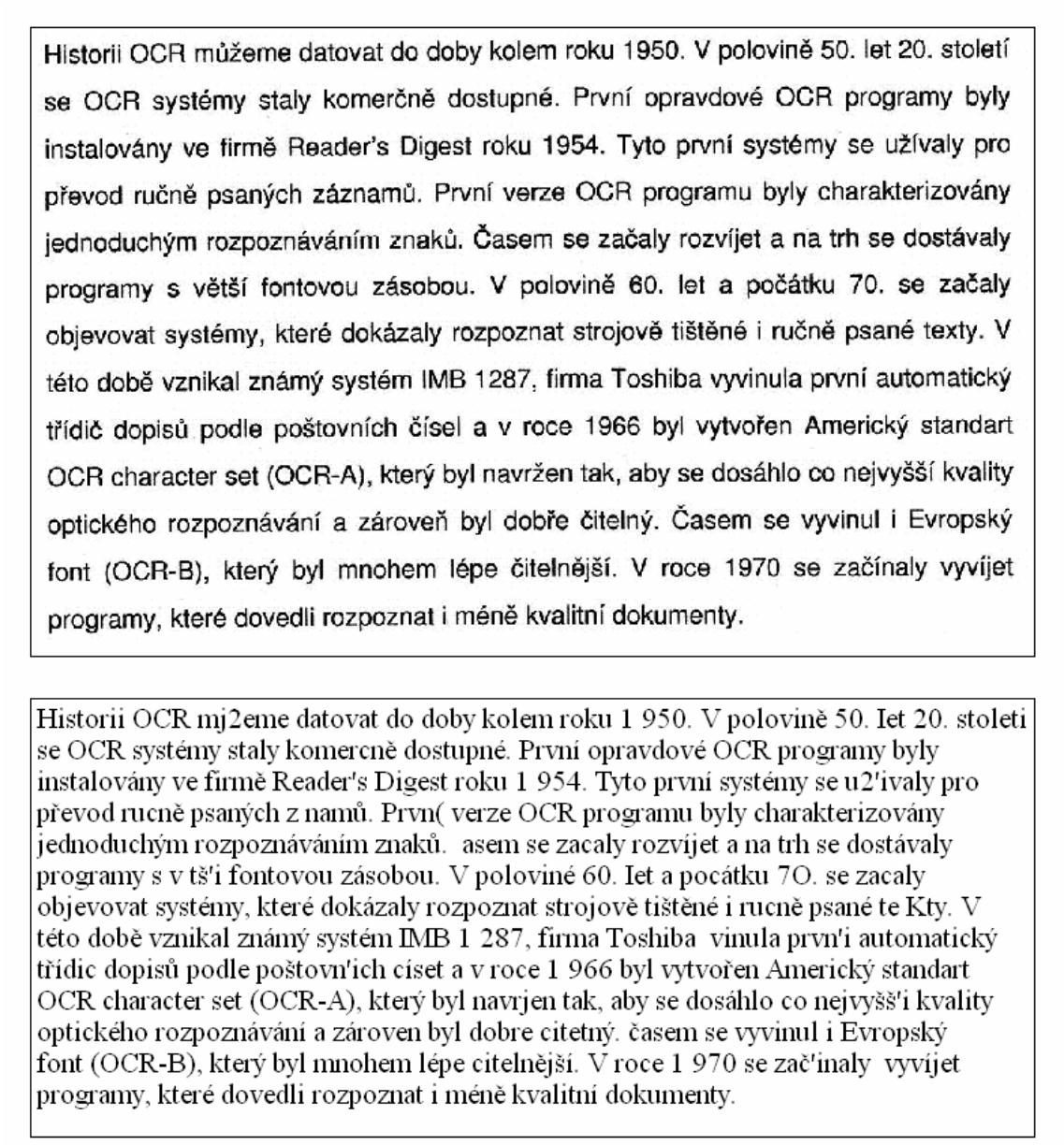


Obrázek 10 Test 2

Poslední testovaný dokument je vytvořený pomocí scanneru. Úspěšnost rozpoznání je: 97,24%

Počet znaků v textu: 1049

Počet špatně převedených znaků: 29



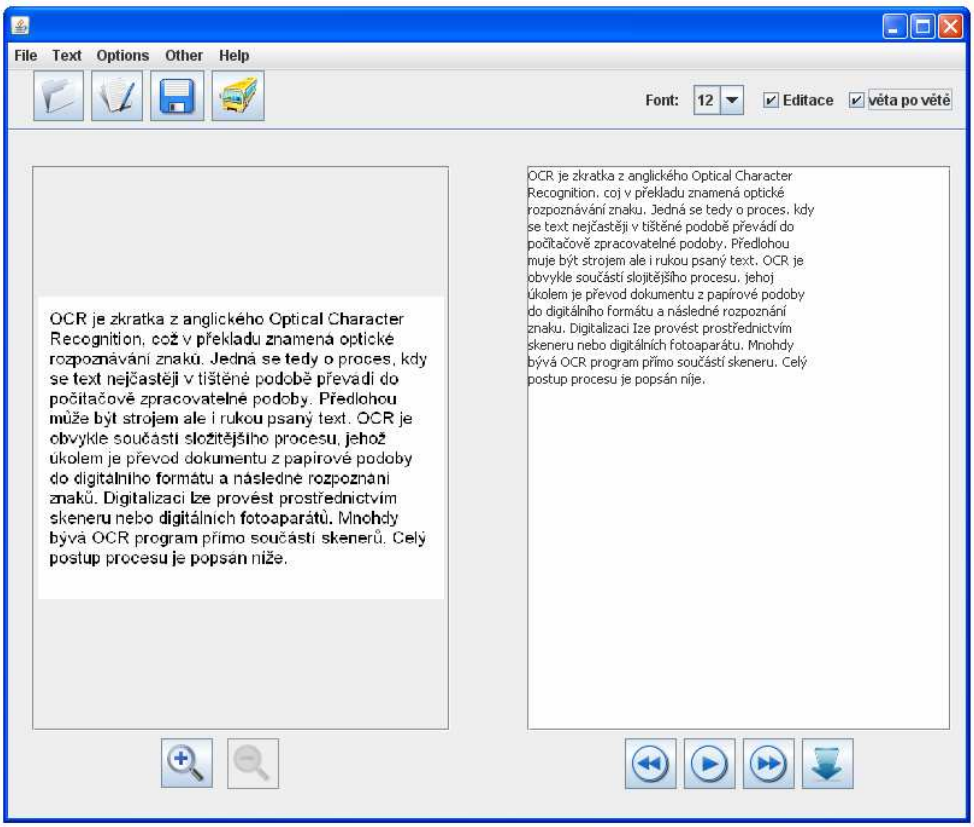
Historii OCR můžeme datovat do doby kolem roku 1950. V polovině 50. let 20. století se OCR systémy staly komerčně dostupné. První opravdové OCR programy byly instalovány ve firmě Reader's Digest roku 1954. Tyto první systémy se užívaly pro převod ručně psaných záznamů. První verze OCR programu byly charakterizovány jednoduchým rozpoznáváním znaků. Časem se začaly rozvíjet a na trh se dostávaly programy s větší fontovou zásobou. V polovině 60. let a počátku 70. se začaly objevovat systémy, které dokázaly rozpoznat strojově tištěné i ručně psané texty. V této době vznikal známý systém IMB 1287, firma Toshiba vyvinula první automatický třídič dopisů podle poštovních čísel a v roce 1966 byl vytvořen Americký standart OCR character set (OCR-A), který byl navržen tak, aby se dosáhlo co nejvyšší kvality optického rozpoznávání a zároveň byl dobře čitelný. Časem se vyvinul i Evropský font (OCR-B), který byl mnohem lépe čitelnější. V roce 1970 se začínaly vyvíjet programy, které dovedli rozpoznat i méně kvalitní dokumenty.

Historii OCR mj2eme datovat do doby kolem roku 1 950. V polovině 50. Iet 20. století se OCR systémy staly komercně dostupné. První opravdové OCR programy byly instalovány ve firmě Reader's Digest roku 1 954. Tyto první systémy se u2'ivaly pro převod rucně psaných z namů. Prvn(verze OCR programu byly charakterizovány jednoduchým rozpoznáváním znaků. asem se zacaly rozvíjet a na trh se dostávaly programy s v tš'i fontovou zásobou. V poloviné 60. Iet a pocátku 7O. se zacaly objevovat systémy, které dokázaly rozpoznat strojově tištěné i rucně psané te Kty. V této době vznikal známý systém IMB 1 287, firma Toshiba vinula prvn'i automatický třídic dopisů podle poštovn'ich císet a v roce 1 966 byl vytvořen Americký standart OCR character set (OCR-A), který byl navrjen tak, aby se dosáhlo co nejvyšš'i kvality optického rozpoznávání a zároven byl dobre citetný. časem se vyvinul i Evropský font (OCR-B), který byl mnohem lépe citelnější. V roce 1 970 se zač'inaly vyvíjet programy, které dovedli rozpoznat i méně kvalitní dokumenty.

Obrázek 11 Test 3

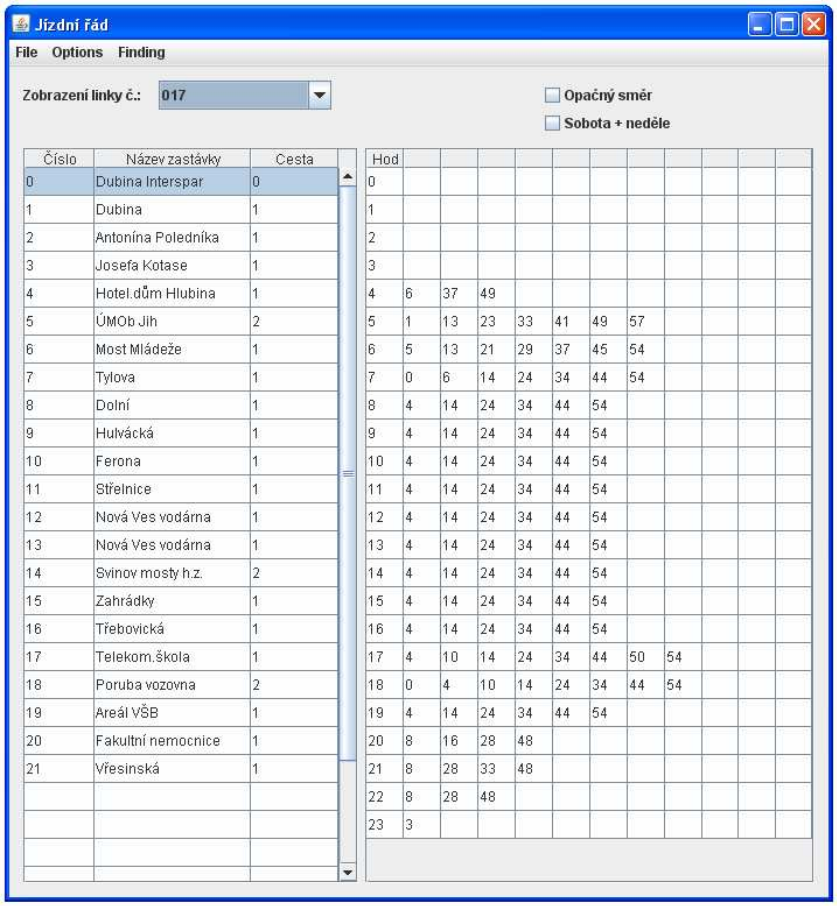
6.9 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní aplikace se skládá z hlavního okna (obr. 11), ve kterém je umístěno hlavní menu a ovládací panel s rychlými tlačítky. Největší část okna zabírají dva panely. Levý panel zobrazuje načtený obrázek či fotografii a v pravém panelu se vypisuje převedený text. Obrázek lze přiblížit a oddálit spodními tlačítky plus a mínus. Převedený text je možno dále editovat. Dále je také zpřístupněna funkce čtení. Existuje několik způsobů přečtení daného textu. A to: funkce věta po větě, čtení aktuální věty (kliknutí kurzoru do věty) a čtení celého textu. K těmto funkcím lze přistupovat tlačítky umístěnými pod textem.



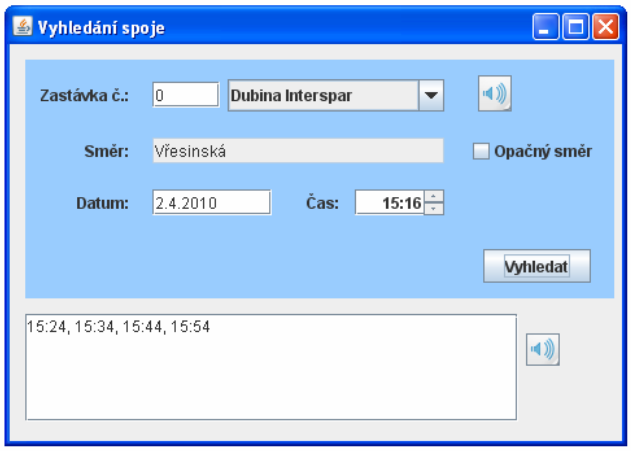
Obrázek 12 Hlavní okno

Další částí této aplikace je elektronický jízdní řád. Tato funkce je umístěna v samostatném okně (obr. 12). Toto okno zobrazuje seznam zastávek dané linky a jednotlivé časy odjezdů v závislosti na zastávce. Lze nastavit opačný směr jízdy i jízdní řády platné o víkendech. Je možné nahrát i více linek a poté mezi nimi jednotlivě přepínat.



Obrázek 13 Jízdní řád

Dále funkce jízdního řádu obsahuje i jednoduché vyhledávání. Toto vyhledávání se zobrazí v samostatném okně, kde se zadávají hodnoty, nutné k vyhledání viz. obr. 13.



Obrázek 14 Vyhledání spoje

7. Závěr

Prostřednictvím této bakalářské práce jsem se seznámila s problematikou optického rozpoznávání znaků, s jeho principem a různými způsoby rozpoznávání. Dále jsem se také okrajově seznámila s problémy slabozrakých a snažila se jim vyjít co nejvíce vstříc v implementaci mé aplikace. Mým původním cílem bylo vyvinout mobilní aplikaci, avšak tento úkol se stal nerealizovatelný, neboť současná nabídka volně stažitelných aplikací pro převod metodou OCR nenabízí vhodné aplikace podporující i mobilní zařízení. Proto jsem zvolila implementaci desktopové aplikace, u které jsem se snažila vytvořit jednoduché a intuitivní grafické uživatelské rozhraní, které by usnadňovalo komunikaci s mým programem. Použitý software Asprise OCR jsem se snažila vylepšit pro české uživatele, používající český text, přidáním metody, která automaticky doplní špatně rozpoznané znaky za znaky s diakritikou a také dále rozepíše zkrácené tvary pro lepší pochopení.

Je nutné však také uvést nedostatky, které by se mohly při používání aplikace vyskytnout. Zvuková stránka aplikace by zasloužila zlepšení v podobě výkonnějšího překladače textu na zvuk, neboť při větším množství překladu, aplikace danou zátěž nezvládá. Další vývoj této aplikace by se měl tedy ubírat jak ve zdokonalení rozpoznávání, tak i v dalším rozvoji zvukové části. Budoucnost vidím zejména ve verzi pro mobilní zařízení, neboť by možnost přečtení textu byla vždy po ruce. Proto by také bylo vhodné zakoupit kvalitní OCR software, podporující i mobilní zařízení.

Literatura

[1] *Data identification* [online]. 2003 [cit. 2010-03-29]. All about OCR. Dostupné z WWW: <http://www.dataid.com/aboutocr.htm>.

[2] BROWN, Eric W. *Character Recognition by Feature Point Extraction* [online]. 1992 [cit. 2010-03-29]. Character Recognition by Feature Point Extraction. Dostupné z WWW: <http://www.ccs.neu.edu/home/feneric/charrec.html>.

[3] TIŠNOVSKÝ, Pavel. *Root.cz* [online]. 7.12.2006 [cit. 2010-03-29]. JPEG - král rastrových grafických formátů?. Dostupné z WWW: <http://www.root.cz/clanky/jpeg-kral-rastrovych-grafickych-formatu/>.

[4] SOBOTA, Bronislav; MILIÁN, Ján. *Grafické formáty*. České Budějovice: Kopp, 1996. 157 s. ISBN 80-85828-58-8.

[5] BOSÁK, Rostislav; FANTA, Martin; PEŘINA, Martin. *Pár kapek Javy* [online]. 1999 [cit. 2010-03-29]. Grafické uživatelské rozhraní a Systém zpráv. Dostupné z WWW: <http://www.ataco.cz/perina/par-kapek/Chapter8/Chap8.html>.

[6] KISZKA, Bogdan. *1001 tipů a triků pro programování v jazyce Java*. Brno: Computer Press, 2003. 519 s. ISBN 80-7226-989-5.

[7] BRETTŠNAJDR, Antonín. *Syntéza zvukového signálu*. [s.l.], 2009. 36 s. Bakalářská práce. Vysoká škola Báňská, Fakulta elektrotechniky a informatiky.

Seznam příloh

A - Obsah CD

Adresář	**Popis**
/src	Zdrojové soubory aplikace
/text	Textové soubory bakalářské práce

B - Uživatelský manuál

Využití aplikace

Tato aplikace je určena zejména lidem zrakově postiženým. Není však podmínkou, aby tuto aplikaci využívali pouze tito lidé. Hlavní funkce této aplikace spočívá v převodu obrazu na text pomocí techniky optického rozpoznávání znaků a následné přečtení. Dodatečnou funkcí této aplikace je jízdní řád, ve kterém je možno vyhledávat spoje. Celá aplikace je opatřena jednoduchým ovládáním. Uživatel si může také vybrat, zda bude ovládat aplikaci pomocí počítačové myši či klávesových zkratek.

Instalace a spuštění aplikace

Instalace aplikace je velmi jednoduchá. Spočívá jen ve dvou krocích.

1. nahrání souboru "AspriseOCR.dll" do složky C:\WINDOWS\system32.
2. spuštění aplikace pomocí souboru "OCR.jar".

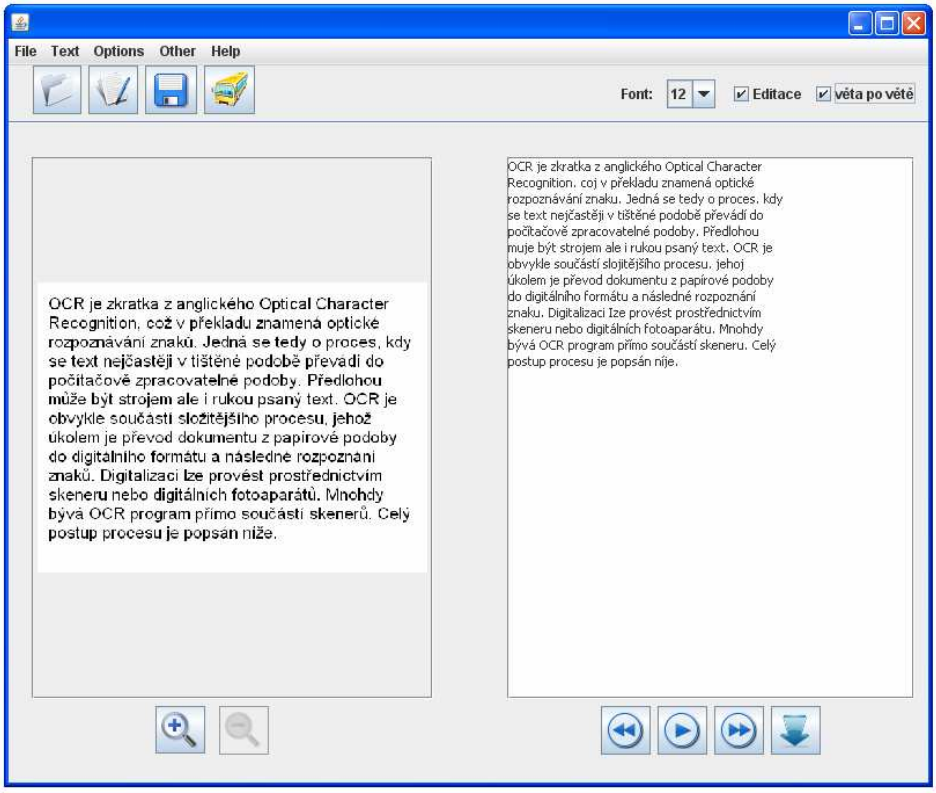
Popis prostředí

Po spuštění aplikace se objeví hlavní okno viz. obr. 1.1. Toto okno se skládá ze tří částí. První částí je lišta menu, která nabízí veškeré ovládání aplikace. Pod jednotlivými položkami se ukrývají funkce otevření obrázku, otevření textového souboru, uložení textového souboru, samotný převod OCR, ale i zvukové ovládání. Menu také umožňuje spustit jízdní řád či okno s nápovědou. Pro snadnější ovládání je pod lištou menu umístěn pruh rychlých voleb, kde jsou k dispozici nejčastěji používaná tlačítka. V pravé části tohoto pruhu se nachází dodatečné ovládání jako je změna velikosti písma, funkce editace či režimu "věta po větě".

Druhou část okna tvoří náhled obrázku. Tato část se nachází v levé části okna. Po načtení obrázku se tento soubor zobrazí v náhledu. Náhled lze zobrazit v původní či zmenšené velikosti.

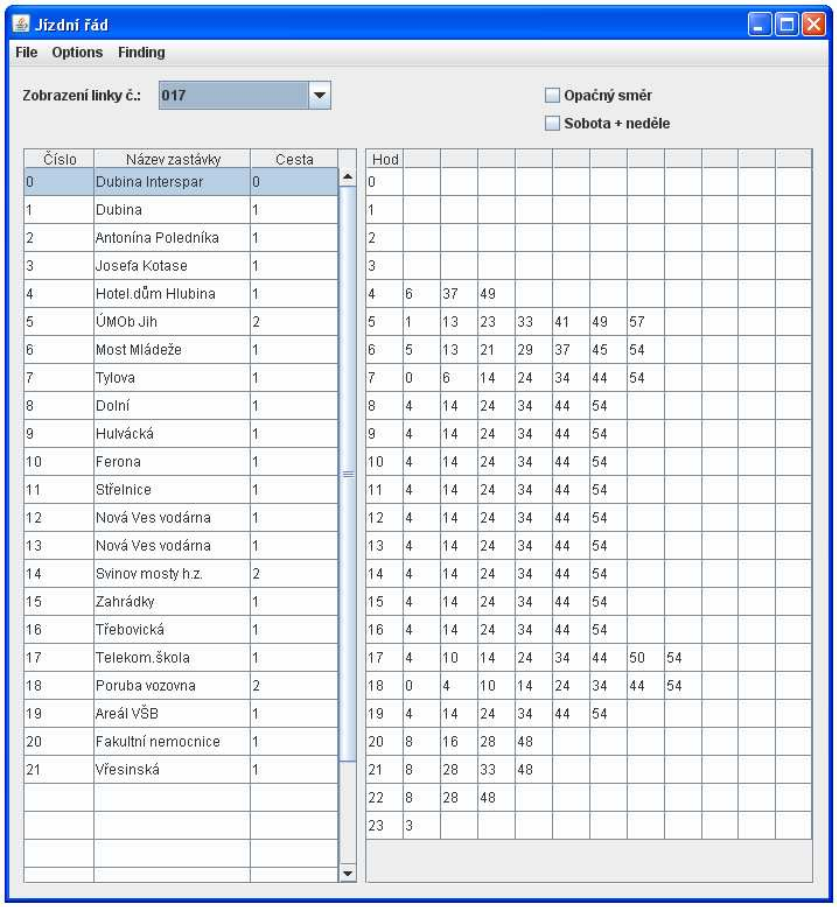
Poslední část tvoří textové pole, ve kterém se zobrazuje převedený či načtený text. Jestliže je povolena volba editace, lze do textového pole zapisovat a upravovat text. Tlačítka

umístěna pod textem umožňují ovládání zvuku. Jednotlivé funkce tlačítek jsou popsány v kapitole Funkce.



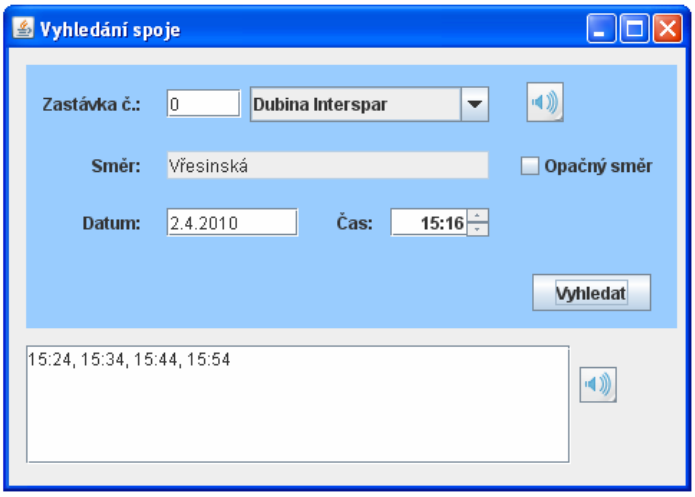
Obr. 1.1 Hlavní okno

Další částí aplikace je funkce jízdního řádu. Jízdní řád má své vlastní okno viz. 1.2. Toto okno má také hlavní menu, ve kterém lze najít veškeré ovládání. Nahraný jízdní řád se zobrazí do okna. V levé části je umístěn seznam zastávek a prává část obsahuje výpis jednotlivých odjezdů.



Obr. 1.2 Jízdní řád

Pro vyhledání spoje se zobrazí nové okno viz. obr. 1.3. Toto okno je tvořeno textovými poli pro vyplnění vstupních údajů. Po stisku tlačítka Vyhledat se zobrazí ve spodní části výpis všech korektních spojů. Tento výpis lze samozřejmě nechat přečíst.



Obr. 1.3 Vyhledání spoje

Funkce

Načtení obrázku

Obrázek lze načíst pomocí rychlého tlačítka Load image  nebo položkou v menu: File -> Load image.

Načtení textového souboru

Textový soubor lze načíst položkou v menu: File -> Load text.

Převedení obrázku na text

K tomuto úkonu lze využít rychlé tlačítko Convert  nebo položku v menu: File -> Convert.

Uložení textu

Převedený text lze uložit pomocí tlačítka  nebo pomocí položky v menu: Text -> Save as...

Editace textu

Text lze editovat pouze tehdy, je-li povolena funkce "Editace". Tuto funkci lze povolit/zakázat pomocí zatržení/odtržení volby Editace, která se nachází v pravé části pruhu pod menu. Jinou možností je najít tuto volbu přímo v menu: Text -> Edit.

Změna fontu

Změnit velikost fontu lze pomocí výběrového pole Font. Toto pole se nachází v pravé části pod menu.

Přiblížení a oddálení obrázku

Zobrazený obrázek lze zobrazit v původní velikosti pomocí tlačítka  a  tlačítkem zase obrázek oddálit.

Ovládání zvuku

Zvuk lze ovládat pomocí čtyř tlačítek umístěných pod textovým polem.

1 2 3 4

1 a 3 - tato tlačítka jsou přístupná pouze tehdy, je-li povolena volba věta po větě. Slouží k přepínání čtených vět.

2 - pomocí tohoto tlačítka program přečte celý text nebo část, která je vyznačena.

4 - pomocí tohoto tlačítka program přečte větu, do které je umístěn kurzor.

Jízdní řád

Spustit tuto funkci lze pomocí položky v menu: Choice -> Jízdní řád nebo ikonou 

Nahrání linky

Novou linku lze nahrát pomocí položky v menu: Linka -> Přidat linku. Otevře se okno s nabídkou nahrání linky buď ze souboru .html uloženého na lokálním disku nebo přímo stažení html stránky z internetu.

Vyhledání spoje

Vyhledání se spustí pomocí položky v menu: Vyhledání. Zobrazí se okno s kolonkami, do kterých je nutno zadat kritéria, podle kterých se bude vyhledávat. Toto tlačítko  slouží pro nadiktování dat.

Seznam klávesových zkratek

Klávesová zkratka	Popis
Ctrl + I	Otevření obrázku
Ctrl + T	Otevření textového dokumentu
C	Převod obrázku na text
Ctrl + X	Ukončení aplikace
Ctrl + E	Editace textu
Ctrl + S	Uložení textového souboru
Ctrl + J	Otevření jízdního řádu
+	Přiblížení náhledu obrázku
-	Oddálení náhledu obrázku
B	Přečtení celého/části textu
M	Přečtení věty označené kurzorem
N	Další věta
V	Předchozí věta
Shift + L	Nahrání nové linky
O	Opačný směr
W	Víkend
F	Otevře okno pro hledání spojení

C - Programátorská dokumentace

Instalace a spuštění aplikace

Instalace aplikace spočívá jen ve dvou krocích.

3. nahrání souboru "AspriseOCR.dll" do složky C:\WINDOWS\system32. Tento soubor je nutný pro správný chod aplikace Asprise OCR, neboť představuje nativní knihovnu této aplikace.

4. spuštění aplikace pomocí souboru "OCR.jar".

Knihovny aplikace

Aplikace využívá knihovnu aplikace Asprise OCR - "aspriseOCR.jar" pro využití metody optického rozpoznávání znaků a knihovny "jdom.jar", "looks-2.2.1.jar" pro práci se zvukem.

Metody

Korekce

Funkce konečné opravy textu je implementována metodou correction(), ve které dochází k nahrazování špatně rozpoznaných znaků a také využívá textového souboru, ve kterém jsou definovány zkratky. Tyto zkratky jsou následně nahrazeny celými slovy.

Převedení

Pro převedení obrazu na text je nutné použít metodu recognition(), ve které je volána metoda knihovny Asprise OCR.

Ozvučení

Přečtení textu se spouští několika metodami, jelikož existuje více možností jak text přečíst. Ve všech těchto metodách se ale využívá metod programu p. Brettšnajdra.

Kompletní popis všech metod, událostí i proměnných je ve vygenerovaném JavaDOCu, který je uložen na CD ve složce /text/JavaDOC.